Leading Open Source Middleware

Ant Tasks for JOnAS User Manual

JOnAS Team (Philippe CoqAdriana Danes,)

- february 2008 -

Table of Contents

1. Introduction 1
2. JOnAS Ant Tasks 2
    2.1. ejbjar Ant Task 2
        2.1.1. Description 2
        2.1.2. Parameters: 2
        2.1.3. Examples 4
    2.2. jonasbase Ant Task 4
    2.3. newjc Ant Task 4
    2.4. jonas Ant Task 4
    2.5. genic Ant Task 5
        2.5.1. Why is there a new task? 5
        2.5.2. Task Attributes 5
        2.5.3. Example: 7
    2.6. wsgen Ant Task 7
    2.7. serverdeploy Ant Task 7
    2.8. property Ant Task 7

Chapter 1. Introduction

Chapter 2. JOnAS Ant Tasks

2.1. ejbjar Ant Task

2.1.1. Description

The `<jonas>` nested element uses the `GenIC` -specific tool to build JOnAS-specific stubs and skeletons and construct a JAR file which may be deployed to the JOnAS Application Server. The build process will always determine if the EJB stubs/skeletons and the EJB-JAR file are up to date, and it will perform the minimum amount of work required.

A naming convention for the EJB descriptors is most commonly used to specify the name for the completed JAR file. For example, if the EJB descriptor `ejb/Account-ejb-jar.xml` is located in the descriptor directory, the `<jonas>` element will search for a JOnAS-specific EJB descriptor file named `ejb/Account-jonas-ejb-jar.xml` , and a JAR file named `ejb/Account.jar` will be written in the destination directory. The `<jonas>` element can also use the JOnAS naming convention. Using the same example, the EJB descriptor can also be named `ejb/Account.xml` (no base name terminator here) in the descriptor directory. The `<jonas>` element will then search for a JOnAS-specific EJB descriptor file called `ejb/jonas-Account.xml` . This convention does not strictly follow the ejb-jar naming convention recommendation, but it is supported for backward compatibility with previous version of JOnAS.

Note that when the EJB descriptors are added to the JAR file, they are automatically renamed `META-INF/ejb-jar.xml` and `META-INF/jonas-ejb-jar.xml` .

Furthermore, this naming behaviour can be modified by specifying attributes in the ejbjar task (for example, basejarname, basenameterminator, and flatdestdir) as well as the iplanet element (for example, suffix). Refer to the appropriate documentation for more details.

2.1.2. Parameters:

Attribute	**Description**	**Required**
destdir	The base directory into which the generated JAR files will be written. Each JAR file is written in directories which correspond to their location within the " `descriptordir` " namespace.	Yes
jonasroot	The root directory for JOnAS.	Yes
jonasbase	The base directory for JOnAS. If omitted, it defaults to `jonasroot.`	No
classpath	The classpath used when generating EJB stubs and skeletons. If omitted, the classpath specified in the "ejbjar" parent task will be used. If specified, the classpath elements will be prefixed to the classpath specified in the parent "ejbjar" task. A nested "classpath" elements can also be used. Note that the needed JOnAS JAR files are	No

Attribute	**Description**	**Required**
	automatically added to the classpath.	
keepgenerated	`true` if the intermediate Java source files generated by GenIC must not be deleted. If omitted, it defaults to `false` .	No
nofastrmic	if `true` , the external RMIC compiler is used by GenIC. The default is `false` , which means the internal fast RMIC compiler is used.	No
nocompil	`true` if the generated source files must not be compiled via the java and rmi compilers. If omitted, it defaults to `false` .	No
novalidation	`true` if the XML deployment descriptors must be parsed without validation. If omitted, it defaults to `false` .	No
javac	Java compiler to use. If omitted, it defaults to the value of `build.compiler` property.	No
javacopts	Options to pass to the java compiler.	No
protocols	Comma-separated list of protocols (chosen within jeremie, jrmp, iiop, cmi) for which stubs should be generated. Default is `jrmp,jeremie` .	No
rmicopts	Options to pass to the rmi compiler.	No
verbose	Indicates whether or not to use -verbose switch. If omitted, it defaults to `false` .	No
additionalargs	Add additional args to GenIC.	No
keepgeneric	`true` if the generic JAR file used as input to GenIC must be retained. If omitted, it defaults to `false` .	No
suffix	String value appended to the JAR filename when creating each JAR. If omitted, it defaults to ".jar".	No
nogenic	If this attribute is set to `true` , JOnAS's GenIC will not be run on the EJB JAR. Use this if you prefer to run GenIC at deployment time. If omitted, it defaults to `false` .	No
jvmopts	Additional args to pass to the GenIC JVM.	No

Attribute	Description	Required
invokecmd	If this attribute is set to `true` , GenIC will use the Javac sun class to avoid using 'javac' command line. This is useful for users getting 'java.io.Exception CreateProcess' because of too long command lines. Defaults to `false` .	No

As noted above, the jonas element supports additional <classpath> nested elements.

2.1.3. Examples

Note : To avoid `java.lang.OutOfMemoryError` , the element `jvmopts` can be used to change the default memory usage.

This example shows ejbjar being used to generate deployment jars using a JOnAS EJB container. This example requires the naming standard to be used for the deployment descriptors. Using this format creates a EJB JAR file for each variation of '*-jar.xml' that is located in the deployment descriptor directory.

```
<ejbjar srcdir="${build.classes}"
        descriptordir="${descriptor.dir}">
  <jonas destdir="${deploymentjars.dir}"
         jonasroot="${jonas.root}"
         protocols="jrmp,iiop"/>
  <include name="**/*.xml"/>
  <exclude name="**/jonas-*.xml"/>
  <support dir="${build.classes}">
       <include name="**/*.class"/>
  </support>
</ejbjar>
```

This example shows ejbjar being used to generate a single deployment jar using a JOnAS EJB container. This example does require the deployment descriptors to use the naming standard. This creates only one ejb jar file - 'TheEJBJar.jar'.

```
<ejbjar srcdir="${build.classes}"
        descriptordir="${descriptor.dir}"
        basejarname="TheEJBJar">
  <jonas destdir="${deploymentjars.dir}"
         jonasroot="${jonas.root}"
         suffix=".jar"
         protocols="${genic.protocols}"/>
  <include name="**/ejb-jar.xml"/>
  <exclude name="**/jonas-ejb-jar.xml"/>
</ejbjar>
```

2.2. jonasbase Ant Task

2.3. newjc Ant Task

2.4. jonas Ant Task

2.5. genic Ant Task

2.5.1. Why is there a new task?

Previous versions of the JOnAS EjbJar Ant task have some limitations, especially when you are developing webservices.

In previous versions of JOnAS, jonas-ejb-jar constructed the JAR for you, using information gathered from the ejb-jar.xml and from the classes themselves (dependencies using BCEL). But if you have a Session Bean exposed as a webservice, you want to have more files in your archive (`webservices.xml` , `JAX-RPC mapping file` , `WSDL` + imported `WSDL definitions` + imported `XML Schema`).

The older task did not package these files inside the archive, and therefore, when GenIC loaded the descriptors, some dependencies were missing, causing GenIC to throw an exception.

The solution is to let the developer create the JAR file, so that the exact content of the file can be controlled.

The file should have at least the following content:

- META-INF/ejb-jar.xml
- META-INF/jonas-ejb-jar.xml
- **/*.class

Notice that webservices files are optional.

2.5.2. Task Attributes

The `genic` task supports most attributes of the `jonas-ejb-jar` task.

Differences:

- Removed `destdir` : JAR files are directly modified
- Removed `classpath` : classpath is now set as inner element
- Removed `novalidation` : validation attribute is used now
- Removed `keepgeneric` : not meaningful (input JAR is already specific)
- Removed `suffix` : not meaningful (no JAR generated)
- Removed `nogenic` : not meaningful (we want to run GenIC)

Attribute	**Description**	**Required**
jonasroot	The root directory for JOnAS.	Yes (Can be read from ${jonas.root} property if not set)
jonasbase	The base directory for JOnAS. If omitted, it defaults to `jonasroot` .	No
keepgenerated	`true` if the intermediate Java source files generated by GenIC must not be deleted. If omitted, it defaults to `false` .	No
nofastrmic	if `true` , the external RMIC compiler is used by GenIC. The	No

Attribute	Description	Required
	default is `false` , which means the internal fast RMIC compiler is used.	
nocompil	`true` if the generated source files must not be compiled via the Java and RMI compilers. If omitted, it defaults to `false` .	No
validation	`true` if the XML deployment descriptors must be parsed with validation. If omitted, it defaults to `true` .	No
javac	Java compiler to use. If omitted, it defaults to the value of `build.compiler` property.	No
javacopts	Options to pass to the Java compiler.	No
protocols	Comma-separated list of protocols (chosen from jeremie, jrmp, iiop, cmi) for which stubs should be generated. Default is `jrmp,jeremie` .	No
rmicopts	Options to pass to the RMI compiler.	No
verbose	Indicates whether or not to use -verbose switch. If omitted, it defaults to `false` .	No
additionalargs	Add additional arguments to GenIC.	No
jvmopts	Additional arguments to pass to the GenIC JVM.	No
jvmdebug	Indicates whether you want to debug the forked JVM; it defaults to `false` . The JVM will be suspended and waiting a connection on port 12345.	No
invokecmd	If this attribute is set to `true` , GenIC will use the Javac sun class to avoid using 'javac' command line. This is useful for users getting 'java.io.Exception CreateProcess' because of too long command lines. Defaults to `false` .	No

Nested Element	Description	Required
classpath	The additional classpath entries used when generating EJB stubs and skeletons.	No
fileset	Points out the ejb-jars that will be processed by GenIC. Note that you can add as many filesets	Yes (at least 1)

Nested Element	Description	Required
	as you want (useful if your JARs are in different directories).	

2.5.3. Example:

1. First, define this new task:

```
<taskdef name="genic"
    classname="org.objectweb.jonas.ant.GenICTask"
    classpath="${jonas.root}/lib/common/ow_jonas_ant.jar" />
```

2. Then use it:

   a. Create the JAR file :

```
<jar destfile="${temp.ejbjars.dir}/my-ejbjar.jar">
    <metainf dir="${etc.dir}/META-INF" />

    <fileset dir="${classes.dir}">
        <include name="org/objectweb/jonas/myejbjar/*.class" />
    </fileset>
</jar>
```

   b. Process this JAR with the GenIC task :

```
<genic keepgenerated="true"
       protocols="${protocols.names}">

    <fileset dir="${temp.dir}">
        <include name="ejbjars/my-ejbjar.jar" />
    </fileset>
</genic>
```

2.6. wsgen Ant Task

2.7. serverdeploy Ant Task

2.8. property Ant Task